Sample Chat Application

Objective:

Implementing a chat application that will populate at startup 200 users in the database and echo back any message sent after a delay of about 500ms.

github.aeharake.choters

This package contains the root Application which is **ApplicationController**

In the **ApplicationController** we initialize Fresco library that will be used in the entire application

github.aeharake.choters.mocker

This includes whatever we would like to mock, we have one class for mocking the 200 users which is: **UsersGenerator**

github.aeharake.choters.activities

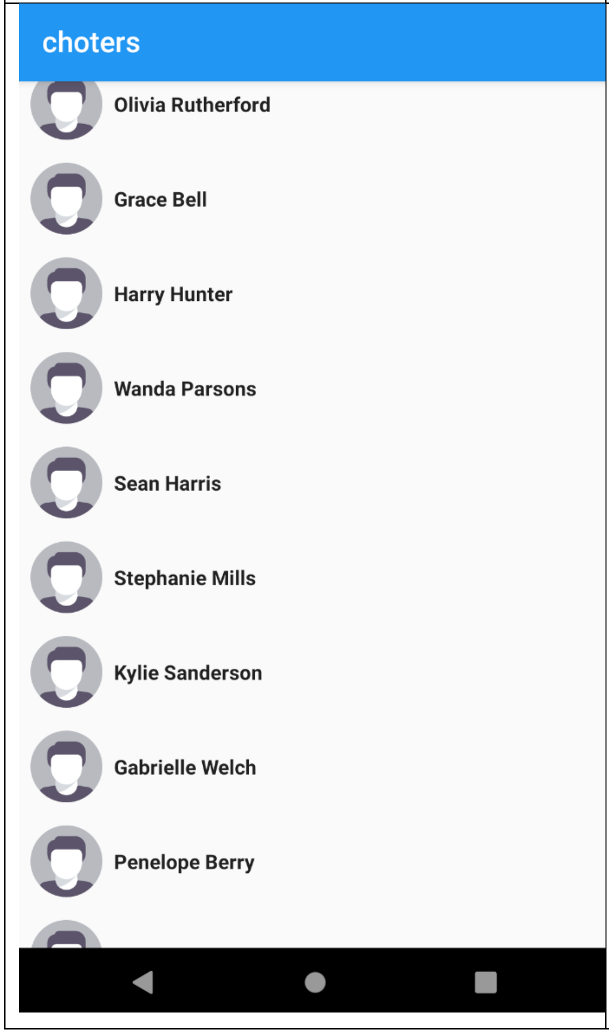
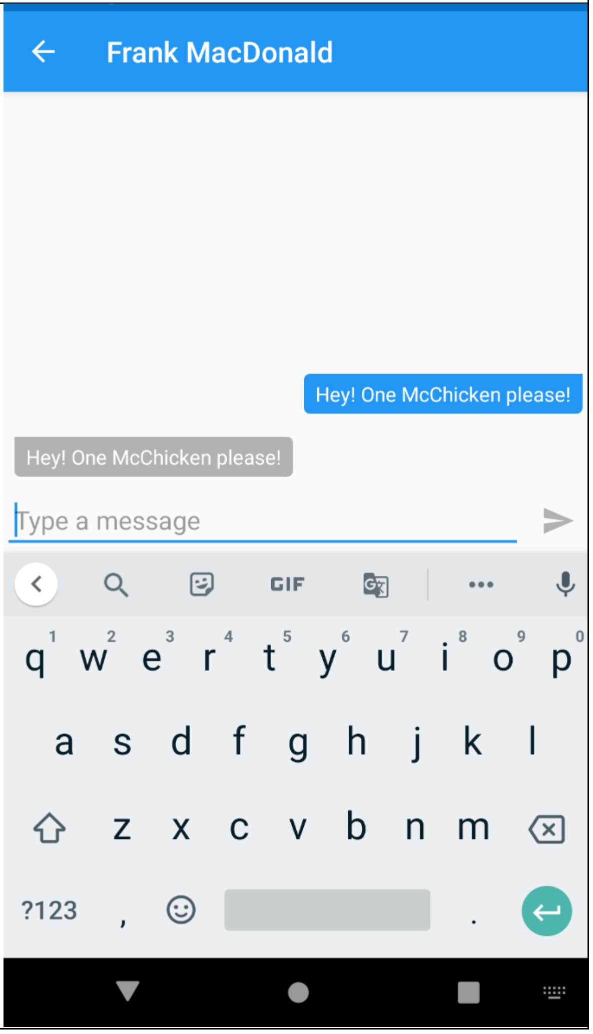
This package contains all the activities. We have two activities:

1. **MainActivity**

   This activity is responsible or showing us the users and their last message

2. **ConversationActivity**

   This activity is responsible for showing us the Conversation between user and his contact.

MainActivity	ConversationActivity
	

choters

Olivia Rutherford

Grace Bell

Harry Hunter

Wanda Parsons

Sean Harris

Stephanie Mills

Kylie Sanderson

Gabrielle Welch

Penelope Berry

Frank MacDonald

Hey! One McChicken please!

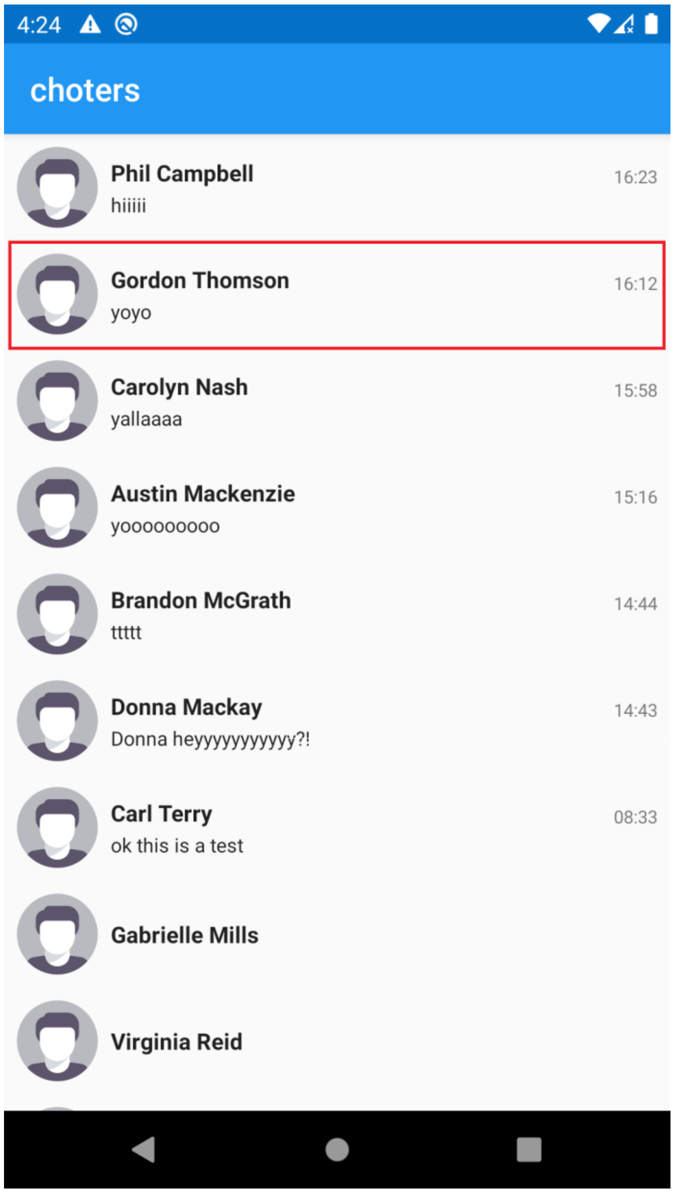
Hey! One McChicken please!

Type a message

github.aeharake.choters.adapters

This package contains all the adapters, we have two adapters:

1. **UsersAdapter**
   This will bind the data to User row in the main activity



4:24

choters

Phil Campbell 16:23
hiiiii

Gordon Thomson 16:12
yoyo

Carolyn Nash 15:58
yallaaaa

Austin Mackenzie 15:16
yooooooooo

Brandon McGrath 14:44
ttttt

Donna Mackay 14:43
Donna heyyyyyyyyyyy?!

Carl Terry 08:33
ok this is a test

Gabrielle Mills

Virginia Reid

2. **ConversationAdapter**
   This will bind the message received/sent in the **ConversationActivity** to **item_row_friend** or **item_row_me**. This is decided from inside the adapter in **getViewType**()

github.aeharake.choters.repos

this package contains all the project repositories. The **BaseRepository** is the parent of all repositories holding reference to the database and the data access object.

We have two repositories in the app:

1. **UsersRepository**

This contains the logic behind:

a. Checking whether the user's table is empty or not. If it's empty then **UsersGenerator** class will be used to generate for us the 200 full names, once we have them we will be creating from this repository our User entity and insert them in bulk using our **UsersDao**
b. Getting the users as **LiveData** so that the **MainActivity** can observe on users changes.

2. **MessageRepository**:

   Responsible for:

   a. Inserting the sent message and the message that should be "echoed" after a certain delay
   b. Getting the messages as live data so that whenever there is a change in the database the recycler-view's adapter should be notified from **ConversationActivity**.

github.aeharake.choters.room

This package contains all what is related to room database, including entities and DAO interfaces:

github.aeharake.choters.room.dao

1. DAO
    a. **BaseDao.kt** (parent interface of all DAO objects)
       This was added to be able to pass it as non-null generic type
    b. **UserDao.kt**
       Responsible for the query to get the users and insert the users
    c. **MessageDao.kt**
       Responsible for the queries of getting the messages between two persons and inserting the messages as well.

github.aeharake.choters.room.entities

2. Entities
    a. **Message.kt**
       This entity has two foreign keys to the User entity, to specify the sender and the receiver
    b. **User.kt**
       This holds the user information
    c. **UserMessage.kt**
       This will holder the user + last message sent/received
3. **ChatDatabase**:

   Responsible for creating the database if it doesn't exist. Only one instance of **ChatDatabase** will be alive during the application's lifecycle

github.aeharake.choters.viewmodels

includes all the view-models of the application

1. **UsersViewModel**
   This will talk with the UsersRepository

   Methods involved:

   fun getAllUsers(): LiveData<List<UserMessage>>

   fun populate()

2. **ConversationsViewModel**
   This will task with the MessageRepository:

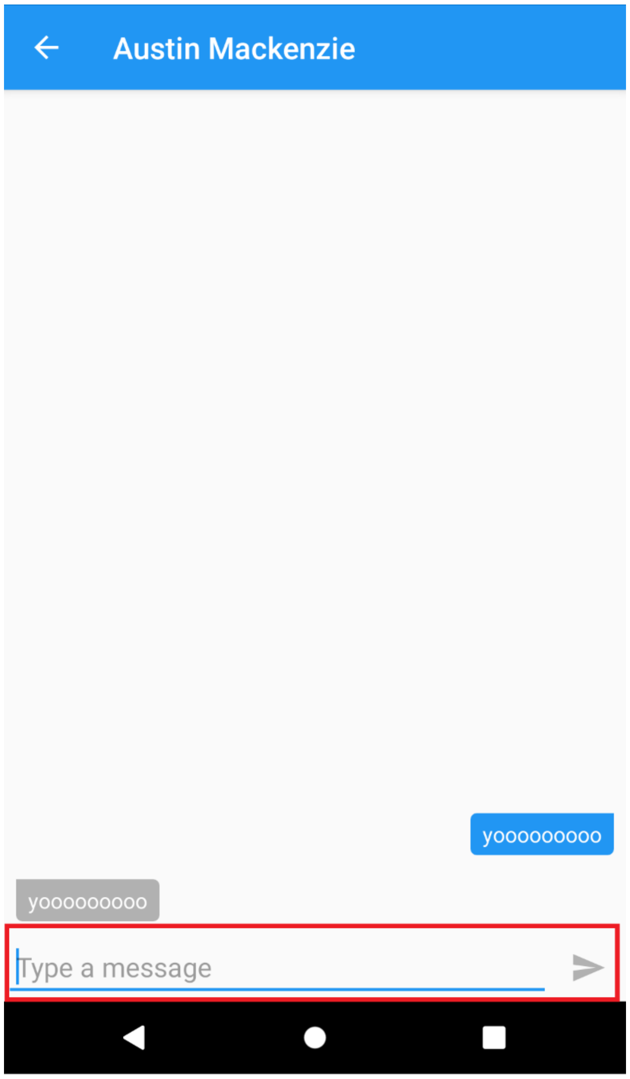
   Methods involved:

   fun getAllMessages(id: Int): LiveData<List<Message>>

   fun insertMessageAndEcho(text: String, id: Int)

github.aeharake.choters.ui

This package contains the custom view which is SendTextView



github.aeharake.choters.utils

Includes the helper classes:

1. **CalendarHelper**
   This class will get us the time in pretty format.

2. **CodeUtils**
   Also, a helper class that holds some code that can be used a lot in the project to avoid code repetition.